IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellants:	Arra E. AVAKIAN et al.	§	Confirmation No.:	1119
		§		
Serial No.:	10/640,620	§	Group Art Unit:	2193
		§		
Filed:	08/12/2003	§	Examiner:	Tuan A. Vu
		§		
For:	Using Interceptors and Out-of-Band Data to Monitor the Performance Of JAVA 2 Enterprise Edition (J2EE)	§ § § § §	Docket No.:	10017134-1

APPEAL BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Date: February 10, 2009

Sir or Madam:

Appellants hereby submit this Appeal Brief in connection with the above-identified application. A Notice of Appeal was electronically filed on December 10, 2008.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is Hewlett-Packard Development Company, L.P. (HPDC), a Texas Limited Partnership, having its principal place of business in Houston, Texas. HPDC is a wholly owned affiliate of Hewlett-Packard Company (HPC). The Assignment from the inventors to HPDC was recorded on February 12, 2004, at Reel/Frame 014976/0795.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals or interferences.

III. STATUS OF THE CLAIMS

Originally filed claims:	1–20.
Claim cancellations:	12–14.
Added claims:	None.
Presently pending claims:	1–11 and 15–20.
Presently appealed claims:	1–11 and 15–20.

IV. STATUS OF THE AMENDMENTS

No claims were amended after the final Office action dated 2008. On December 9, 2008, the Specification was amended and a Terminal Disclaimer was filed. Per an Advisory Action dated January 9, 2009, the proposed amendments to the Specification were entered.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

This section provides a concise explanation of the subject matter defined in each of the independent claims and separately argued dependent claims involved in the appeal, referring to the specification by page and line number or to the drawings by reference characters as required by 37 C.F.R. § 41.37(c)(1)(v). Each element of the claims is identified with a corresponding reference to the specification or drawings where applicable. Note that the citation to passages in the specification or drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element. Also note that these specific references are not exclusive; there may be additional support for the subject matter elsewhere in the specification and drawings.

The current application discloses methods and systems for performance monitoring of transactions that originate from a computer system, for example, from a user's desktop computer. The transactions can invoke one or more transactions in other computer systems such as web, application, and database servers. Such transactions typically return data back to the originating computer. For example, the transaction can be a database query initiated by a web browser and processed by a number of software components running on various servers providing a chain of communication between the web browser and the database. More particularly, the invention installs transaction monitoring agents, or "hooks," on one or more of these servers to monitor performance of any of the transactions. As an example, one performance indicator that may be monitored is time required for execution. These performance indicators are helpful to alert computer administrators, computer programmers, and website developers to the source of underperformance in a network of multiple computers.

Independent claim 1 recites a method for monitoring performance of a plurality of transactions in a Java 2 Enterprise Edition ("J2EE") application server including a top level transaction p.32, ℓ.29–p.33, ℓ.4 and plurality of transactions relating to the top level transaction in a child parent hierarchy p.32, ℓ.29–p.33, ℓ.4. For each of the selected plurality of transactions, a performance metric

corresponding to the selected transaction is obtained p.7, ℓ.22–25. Additionally, an instrument hook is installed prior to execution of the selected transaction p.10, ℓ.11–12; p.12, ℓ.3–4; p.30, ℓ.42–43; p.18, ℓ.20–25; p.18, ℓ.1–p.21, ℓ.9; Fig.3 refs.26, 28. The selected transaction is instrumented upon execution using one or more plug-in instruments called by the instrument hook p.10, ℓ.16–18; p.11, ℓ.18–21; p.21, ℓ.10–p.24, ℓ.10. Also, the top level transaction is initiated in response to a request received from a web server p.32, ℓ.25–27; Fig.17 refs.60, 62, 17. A cookie is transmitted from the web server to the application server together with the request p.37, ℓ.7–17; p.44, ℓ.19–20. Correlators for generated for identifying the top level transaction and a parent transaction, if any, upon execution of each of the instrumented transactions p.33, ℓ.12–14; p.33, ℓ.15–p.35, ℓ.21. The cookie is utilized to generate the correlator identifying the top level transaction p.37, ℓ.7–17. The correlators are utilized to cross-correlate a performance metric corresponding to a parent transaction with one or more performance metrics corresponding to one or more child transactions of the parent transaction p.32, ℓ.7–10; p.34, ℓ.16-18. The one or more plug-in instruments implement an interface that communicates data for the performance metric p.11, ℓ.29–31; p.9, ℓ.19–23.

Independent claim 15 recites a method for monitoring performance of at least two Java transactions that are related to one another as parent-child transactions p.32, ℓ.29–p.33, ℓ.4. The method comprises obtaining a performance metric corresponding to each of the at least two Java transactions p.7, ℓ.22–25. An instrument hook is installed prior to execution of each of the at least two Java transactions p.10, ℓ.11–12; p.12, ℓ.3–4; p.30, ℓ.42–43; p.18, ℓ.20–25; p.18, ℓ.1–p.21, ℓ.9; Fig.3 refs.26, 28. A top level transaction of the at least two Java transactions is initiated in response to a request received from a web server p.32, ℓ.25–27; Fig.17 refs.60, 62, 17. The web server transmits a cookie to an application server together with the request p.37, ℓ.7–17; p.44, ℓ.19–20. Additionally, each of the at least two Java transactions is instrumented upon execution using one or more plug-in instruments called by the instrument hook p.10, ℓ.16–18; p.11, ℓ.18–21; p.21, ℓ.10–p.24, ℓ.10. The method further comprises generating a correlator corresponding to the parent transaction p.33, ℓ.12–14;

p.33, ℓ.15–p.35, ℓ.21. Also, RMI over IIOP is utilized to send the parent correlator incorporated in a header of an IIOP message to the child transaction upon execution p.37, ℓ.24–p.38, ℓ.3. Another correlator corresponding to the child transaction is generated p.33, ℓ.27–31, and an additional correlator corresponding to the top level transaction is generated utilizing the cookie p.37, ℓ.7–17. The one or more plug-in instruments implement an interface that communicates data for the performance metric p.11, ℓ.29–31; p.9, ℓ.19–23.

Independent claim 20 recites a computer readable medium comprising instructions operable by a computer which when executed causes the computer to perform a method p.6, ℓ.1–p.7, ℓ.25. The method comprises obtaining a performance metric corresponding to a selected transaction of a plurality of parent-child transactions p.7, ℓ.22–25. An instrument hook is installed prior to execution of the selected transaction p.10, ℓ.11–12; p.12, ℓ.3–4; p.30, ℓ.42–43; p.18, ℓ.20–25; p.18, ℓ.1–p.21, ℓ.9; Fig.3 refs.26, 28. The selected transaction is instrumented upon execution of the selected transaction using one or more plug-in instruments called by the instrument hook p.10, ℓ.16–18; p.11, ℓ.18–21; p.21, ℓ.10–p.24, ℓ.10. Correlators are generated for each of the transactions, and each correlator identifies a top level transaction and a parent transaction p.33, ℓ.12–14; p.33, ℓ.15–p.35, ℓ.21; p.37, ℓ.7–17; if any, corresponding to its associated transaction. The top level transaction is initiated in response to a request received from a web server p.32, ℓ.25–27; Fig.17 refs.60, 62, 17, and the web server transmits a cookie to an application server together with the request p.37, ℓ.7–17; p.44, ℓ.19–20. The cookie is utilized to generate the correlator for the top level transaction p.37, ℓ.7–17. One or more plug-in instruments implement an interface that communicates data for the performance metric p.11, ℓ.29–31; p.9, ℓ.19–23.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Whether claims 1–11 and 20 are unpatentable over Labadie et al. (U.S. Pub. No. 2003/0195959, hereinafter "Labadie") in view of Hind et al. (U.S. Pat. No. 7,003,565, hereinafter "Hind").

Whether claims 15–19 are unpatentable over Labadie and Hind in view of Bansal et al. (U.S. Pub. No. 2003/0120593, hereinafter "Bansal").

VII. ARGUMENT

The claims do not stand or fall together. Instead, Appellants present separate arguments for various independent and dependent claims. After a brief discussion of the cited art, each of these arguments is separately argued below and presented with separate headings as required by 37 C.F.R. § 41.37(c)(1)(vii).

A. Discussion of the Cited Art—Labadie (Pub. No. 2003/0195959)

Labadie discloses methods and software to provide correlators over multiple computers to harmonize the logs of the computers ¶ 2–7; claims 1, 18, 30. Accordingly, Labadie teaches an execution thread having a logging data class defining a data structure for distributed logging Figs. 4A–C, 5A; ¶ 61. A provider plug-in defines a provider service, which may be a logging service ¶ 34. A middleware object between two client objects and a provider class 310 implements the logging service to log events between client objects Labadie ¶ 59; Figs.2, 4A–C, and 5A–B; ¶¶ 61 and 34–35.

A server-side DCS Middleware plug-in obtains correlators from the server-side DCS service for transport to the client over network ¶¶ 32–34. The client transports correlators to server as well Fig.2; ¶¶ 32–34. The correlator data structure has a means for defining an association between the correlator data structure and a partner correlator data structure ¶ 13. The "ARM correlators provide identification only to the level of a transaction instance" ¶¶ 5–6.

The correlators are generated and transported using Simple Object Access Protocol ("SOAP") parameters ¶ 73. Furthermore, with regard to the Tivoli ARM application, Labadie teaches that "component transactions [are] referred to as the children of the parent transaction" and that "each application responsible for a component . . . is modified to include calls" that "may request correlators" ¶ 5.

B. Obviousness rejection of Claims 1–11 and 20

Independent claim 1 recites, in part, "installing an instrument hook prior to execution of the selected transaction." Independent claim 20 recites a similar limitation. However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the quoted limitation. Examiner cites figures 4A-C, 5A, and ¶ 61 of Labadie as allegedly teaching the quoted limitation. At the

locations cited by Examiner, Labadie teaches an execution thread having a logging data class 330 defining a data structure for distributed logging. However, defining a method for distributed logging does not teach or suggest "installing an instrument hook prior to execution of the selected transaction." Examiner responds on page 15 of the Final Office Action: "The rejection has identified ARM methodology including instrumentation having source data, class structure based on which correlators, counter, and identifiers are defined to effectuate a instrumenting of transactional performance/behavior and threads spawned as a result of a runtime. The 'prior to' execution time is clearly taught." Appellants respectfully disagree. Instrumenting transactional performance as a result of runtime does not teach or suggest installing an instrument hook prior to execution. Additionally, Hind does not satisfy the deficiencies of Labadie. For at least this reason, Examiner erred in rejecting independent claims 1 and 20 along with dependent claims 2–11.

Additionally, independent claim 1 recites, in part, "instrumenting said selected transaction . . . using one or more plug-in instruments called by the instrument hook." Independent claim 20 recites a similar limitation. However, Examiner erred in rejecting the claims because the cited reference fails to teach or suggest the limitations. Examiner cites figures 2, 4A–C, 5A–B and ¶¶ 61 and 34–35 of Labadie as allegedly teaching the quoted limitation. At the locations cited by Examiner, Labadie teaches a provider plug-in 152 defining a provider service, which may be a logging service. (Labadie ¶ 34). Furthermore, Labadie teaches a middleware object between two client objects and a provider class 310 implementing a distributed logging service to log events between client objects. (Labadie ¶ 59). However, a provider plug-in defining a logging service does not teach or suggest that the instrument hook calls a plug-in instrument in order to instrument a transaction, as required by the quoted limitation. Examiner responds on page 15 of the Final Office Action:

> The 'plug-in instrument' can be interpreted as a instance of dynamically created API to implement a class purported a instrumentation service, such that such class invokes the creation or constructing of the API. The Tivoli service invokes a service of

> logging provided in form of plug-in DCS which in turn creates *interfaces* or API (see Labadie: *class 310, implements provider class* – para 0059, pg.5) implementing a provider class in order to execute instrumentation such as logging of events by this DCS. This 'plug-in instrument' is construed as the dynamically created interface (or API) to implement this *provider class* of said DCS service; wherein the DCS class provider reads on a instrumentation type ARM or runtime hook invoking said plug-in class/interface; i.e. this provider class viewed as a instrumentation hook or instrument object in view of the correlator needed within ARM-related API in Tivoli, regarding which the DCS classes acts a middleware object (e.g. for logging hook) between high level server and client data (see Labadie: para 0035, pg.4; ARM API, para 0005, pg. 1); and this is deemed satisfying the above limitations . . . .

Appellants respectfully disagree. The interpretation and construction Examiner proposes, even if reasonable, does not teach or suggest the claim limitation. An API that implements the provider class of a DCS service does not teach or suggest instrumenting the selected transaction using one or more plug-in instruments called by the instrument hook. Additionally, Hind does not satisfy the deficiencies of Labadie. For at least this additional reason, Examiner erred in rejecting independent claims 1 and 20 along with dependent claims 2–11.

Additionally, independent claim 1 recites, in part, "initiating said top level transaction in response to a request received from a web server." Independent claim 20 recites a similar limitation. However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the limitations. Examiner cites figure 2 and ¶¶ 32–34 of Labadie as allegedly teaching the quoted limitation. At the locations cited by Examiner, Labadie teaches a server-side DCS Middleware plug-in 146 obtaining correlators from the server-side DCS service 144 for transport to the client 122 over network 142. (Labadie ¶¶ 32–34). The cited location also teaches the converse (*i.e.*, client transporting correlators to server). However, the transmission of partner correlators between client and server over a network does not teach or suggest "initiating said top level transaction in response to a request received from a web server." Examiner responds on page 16 of the Final Office Action: "[Appellants'] arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that

the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the reference." Appellants respectfully disagree. Appellants argue that the claim language "initiating said top level transaction in response to a request received from a web server" comprises patentably distinguishable subject matter because it is not taught or suggested by Examiner's citations to references. It is not taught or suggested by Examiner's citations because citations to of partner correlators between client and server over a network do not teach or suggest initiating said top level transaction in response to a request received from a web server. Additionally, Hind does not satisfy the deficiencies of Labadie. For at least this additional reason, Examiner erred in rejecting independent claims 1 and 20 along with dependent claims 2–11.

Additionally, independent claim 1 recites, in part, "generating correlators for identifying said top level transaction and a parent transaction." Independent claim 20 recites a similar limitation. However, Examiner erred in rejecting the claims because the cited references fail to teach or suggest the limitations. Examiner cites ¶¶ 5 and 12–13 of Labadie as allegedly teaching the quoted limitation. At the locations cited by Examiner, Labadie teaches a correlator data structure having a means for defining an association between the correlator data structure and a partner correlator data structure. (Labadie ¶ 13). Examiner relies on the ARM application as teaching the parent-child relationship, and Labadie further teaches that "ARM correlators provide identification only to the level of a transaction instance." (Labadie ¶¶ 5–6). However, an association between partner-level correlators fails to teach or suggest "generating correlators for identifying said top level transaction and a parent transaction." Examiner responds on pages 16 and 17 of the Final Office Action:

> The hierarchy of parent child transaction has been shown in Labadie, and the ARM to utilize the needed correlator via the DCS invocation and instantiating of interfaces would be sufficient to disclose that correlator is for identifying a identification of a parent name or child name, and this has been taught in the instrumentation portions and the very nature of correlating a parent transaction to its child transaction . . . .

Appellants respectfully disagree. The showing of a hierarchy of a parent-child transaction along with DCS invocation and instantiation does not teach generating correlators for identifying said top level transaction and a parent transaction. Additionally, Hind does not satisfy the deficiencies of Labadie. For at least this additional reason, Examiner erred in rejecting independent claims 1 and 20 along with dependent claims 2–11.

Additionally, independent claim 1 recites, in part, "utilizing said correlators to cross-correlate a performance metric corresponding to a parent transaction with one or more performance metrics corresponding to one or more child transactions of said parent transaction." However, Examiner erred in rejecting the claim because the cited references fail to teach or suggest the limitation. Examiner cites figures 5B, 6A-C, and ¶ 73 of Labadie as allegedly teaching the quoted limitation. At the locations cited by Examiner, Labadie teaches generating and transporting correlators including a SOAP parameter. (Labadie ¶ 73). Furthermore, with regard to the Tivoli ARM application, Labadie teaches that "component transactions [are] referred to as the children of the parent transaction" and that "each application responsible for a component . . . is modified to include calls" that "may request correlators." (Labadie ¶ 5). Thus, Labadie only teaches that children (*i.e.*, component transactions) request correlators. However, children requesting correlators fails to teach or suggest cross-correlation of "a parent transaction." Examiner responds on page 17 of the Final Office Action: "The correlator-based instrumentation is implemented to track transaction related parameters in the bi-directional flowing of events in the paradigm wherein parent transaction invokes a child transaction." Appellants respectfully disagree. While a parent transaction may invoke a child transaction, it does not necessarily flow that the performance metrics between parent and child are cross-correlated. Additionally, Hind does not satisfy the deficiencies of Labadie. For at least this additional reason, Examiner erred in rejecting independent claim 1 along with dependent claims 2–11.

C. Obviousness Rejection of Claims 15–19

Independent claim 15 recites three limitations similar to independent claim 1: "installing an instrument hook prior to execution of each of said at least two Java transactions," "instrumenting each of said at least two Java transactions upon execution . . . using one or more plug-in instruments called by the instrument hook," and "a top level transaction of the at least two Java transactions is initiated in response to a request received from a web server." Examiner also cited Labadie against these limitations. Additionally, neither Hind nor Bansal satisfy the deficiencies of Labadie. Therefore, the reasoning above applies, and Examiner erred in rejecting independent claim 15 along with dependent claims 16–19 for at least these three reasons.

D. Conclusion

For the reasons stated above, Appellants respectfully submit that the rejections should be reversed. In the course of the foregoing discussions, Appellants may have at times referred to claim limitations in shorthand fashion, or may have focused on a particular claim element. This discussion should not be interpreted to mean that the other limitations can be ignored or dismissed, or that limitations from the specification can be imported into the claims. The claims must be viewed as a whole, and each limitation of the claims must be considered when determining the patentability of the claims. Moreover, it should be understood that there may be other distinctions between the claims and the prior art which have yet to be raised, but which may be raised in the future.

It is believed that no extensions of time or fees are required, beyond those that may otherwise be provided for in documents accompanying this paper. However, in the event that additional extensions of time are necessary to allow consideration of this paper, such extensions are hereby petitioned under 37 C.F.R. § 1.136(a), and any fees required (including fees for net

addition of claims) are hereby authorized to be charged to Hewlett-Packard Development Company's Deposit Account No. 08-2025.

Respectfully submitted,

/Tim D. Chheda/

Tim D. Chheda
PTO Reg. No. 60,752
CONLEY ROSE, P.C.
(713) 238-8000 (Phone)
(713) 238-8008 (Fax)
ATTORNEY FOR APPELLANTS

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
Legal Dept., M/S 35
P.O. Box 272400
Fort Collins, CO 80527-2400

VIII. CLAIMS APPENDIX

1. (Previously presented) In a J2EE application server, a method for monitoring performance of a plurality of transactions including a top level transaction and plurality of transactions relating to said top level transaction in a child parent hierarchy, comprising, for each of selected ones of said plurality of transactions, obtaining a performance metric corresponding to the selected transaction by:

installing an instrument hook prior to execution of the selected transaction;

instrumenting said selected transaction upon execution of the selected transaction using one or more plug-in instruments called by the instrument hook;

initiating said top level transaction in response to a request received from a web server;

transmitting a cookie from said web server to said application server together with said request;

generating correlators for identifying said top level transaction and a parent transaction, if any, upon execution of each of said instrumented transactions;

utilizing said cookie to generate the correlator identifying said top level transaction; and

utilizing said correlators to cross-correlate a performance metric corresponding to a parent transaction with one or more performance metrics corresponding to one or more child transactions of said parent transaction,

wherein the one or more plug-in instruments implement an interface that communicates data for the performance metric.

2. (Previously presented) The method of claim 1, wherein the step of instrumenting said selected transaction comprises inserting instrumentation code in a bytecode representation of said selected transaction.

3. (Previously presented) The method of claim 2, wherein said performance metric corresponds to a response time of said selected transaction.

4. (Previously presented) The method of claim 3, wherein said instrumentation code effects generation of a start time marker upon start of execution of said selected transaction and generation of a stop time marker upon completion of execution of said selected transaction.

5. (Original) The method of claim 4, wherein said instrumentation code generates calls to an Application Response Measurement (ARM) agent to cause generation of said stop and start time markers.

6. (Previously presented) The method of claim 5, further comprising utilizing said start and stop time markers to measure a response time of said selected transaction.

7. (Previously presented) The method of claim 1, further comprising generating a record for each instrumented transaction upon completion of said instrumented transaction, said record indicating said performance metric associated with said instrumented transaction, a parent of said instrumented transaction, and said top level transaction.

8. (Previously presented) The method of claim 7, further comprising transmitting said instrumented transaction record to an analysis and presentation module.

9. (Original) The method of claim 1, further comprising storing said correlators in a thread local storage stack in case of execution of said hierarchical transactions in a single thread.

10. (Original) The method of claim 9, further comprising storing said correlators in the stack based on a LIFO protocol.

11. (Previously presented) The method of claim 10, further comprising removing one correlator of said correlators from said stack upon completion of said hierarchical transaction associated with said correlator.

15. (Previously presented) A method for monitoring performance of at least two Java transactions that are related to one another as parent-child transactions, comprising obtaining a performance metric corresponding to each of said at least two Java transactions by:

installing an instrument hook prior to execution of each of said at least two Java transactions, wherein a top level transaction of the at least two Java transactions is initiated in response to a request received from a web server and said web server transmits a cookie to an application server together with said request; and

instrumenting each of said at least two Java transactions upon execution of each of said at least two Java transactions using one or more plug-in instruments called by the instrument hook;

generating a correlator corresponding to said parent transaction, utilizing RMI over IIOP to send said parent correlator incorporated in a header of an IIOP message to said child transaction upon execution;

generating another correlator corresponding to said child transaction; and

generating an additional correlator corresponding to the top level transaction utilizing said cookie, and

wherein the one or more plug-in instruments implement an interface that communicates data for the performance metric.

16. (Previously presented) The method of claim 15, further comprising employing said correlators to cross correlate the performance metric of said parent transaction with the performance metric of said child transaction.

17. (Previously presented) The method of claim 15, wherein said performance metric corresponds to a response time of each of said at least two Java transactions.

18. (Previously presented) The method of claim 17, wherein said performance metric corresponds to a start time marker upon start of execution of each of said at least two Java transactions and a stop time marker upon completion of execution of each of said at least two Java transactions.

19. (Original) The method of claim 18, wherein said modified bytecode representation generate calls to an Application Response Measurement (ARM) agent to cause generation of said start and stop time markers.

20. (Previously presented) A computer readable medium comprising instructions operable by a computer which when executed causes the computer to perform a method comprising:

obtaining a performance metric corresponding to a selected transaction of a plurality of parent-child transactions by

installing an instrument hook prior to execution of the selected transaction; and

instrumenting said selected transaction upon execution of the selected transaction using one or more plug-in instruments called by the instrument hook; and

generating correlators for each of said transactions, wherein each correlator identifies a top level transaction and a parent transaction, if any, corresponding to its associated transaction, wherein said top level transaction is initiated in response to a request received from a web server, and wherein said web server transmits a cookie to an application server together with said request,

utilizing said cookie to generate said correlator for said top level transaction, and

wherein the one or more plug-in instruments implement an interface that communicates data for the performance metric.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.